Public Comments on NIST Draft Special Publication 800-56C

NIST received the following public comments on the draft Special Publication 800-56C, "*Draft Recommendation for Key Derivation through Extraction-then-Expansion" (September 2010)*".

The comments are ordered based on the dates they were received. Most comments were received in e-mail format. The line-breaks are deleted when copied to this file. The e-mail headers are removed to protect commenter's privacy. For the same reason, e-mail addresses, postal mailing addresses, and telephone numbers are also removed from the signatures.

Roger Khazan

A quick comment: consider stating explicitly that there is no secrecy assumption on salt s. Salt s is used as a key in HMAC, so we need to make it clear that s is assumed to be derived from nonces that are exchanged in the clear.

Roger

Nikolajs Volkovs

Correction on the original comments (Second e-mail)

Please, in the comment, instead of

As on the stage of the Key Expansion we use a bigger key, this allows "keying" the "bigger part" of the space of the inner states of the function and, it seems this increasing the "unknowable" level of the function.

read

As on the stage of the Key Expansion we use a bigger key, this allows "keying" the "bigger part" of the space of the inner states of the function and, this increasing the vagueness (or uncertainty) of the function.

Thank you,

Nikolajs Volkovs

Original comments (First e-mail)

Dear NIST,

I would like to propose the following Key Derivation through Extraction-then-Expansion scheme that is based on the ERINDALE-PLUS hashing algorithm - PCT/CA2009/001093

As the number of inner states of the ERINDALE-PLUS function can be huge (greater than 2^{15000}), can be varied, and the construction of the algorithm allows performing very fast switching between the inner states of the function, this gives the possibility to construct a new methodology of secure hashing. Instead of applying a key to a hashed message, we can 'keying' the inner state of the function. Hashing a message with such "keyed" function is like hashing a message with unknown hash function or, more precisely, with the hash function with unknown inner structure.

Such secure hashing solution has a few very important and unexpected properties.

Firstly, the size of a key is not related anymore to the size of a hash value as it takes place, for instance in the case of HMAC, as the size of a key now is related to the size of the space of inner states of the ERINDALE-PLUS hash function. If we consider a hash function, say, with 2^{768} inner states, the size of the key can be up to 768 bits regardless of the size of a hash value generated by the function.

The Key Derivation through Extraction-then-Expansion scheme that is based on the ERINDALE-PLUS could be realized in the following way.

Firstly, K_DK should have the size larger than the size of K_M. The best variant is to generate 512 bits K_DK, regardless of the size of K_M.So, on the stage of the Randomness extraction we use an ERINDALE-PLUS function that generates, say, 512 bits output.

Then, on the stage of the Key Expansion, K_DK is used as a key for the ERINDALE-PLUS function (the same that has been used for generating K_DK), however the parameter that defines the length of the output is adjusted, that is, the needed length (the length of K_M) is set. The ERINDALE-PLUS allows using the length of an output as a parameter. With K_KD as a key and the corresponding input we generate K_M of desired length.

As on the stage of the Key Expansion we use a bigger key, this allows "keying" the "bigger part" of the space of the inner states of the function and, this increasing the vagueness (or uncertainty) of the function.

Sincerely,

Nikolajs Volkovs

Hugo Krawczyk

Issue 1: RFC 5869
=================

I believe that it is VERY IMPORTANT that 56C provides HKDF from RFC 5869 "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)" (http://tools.ietf.org/html/rfc5869) as an explicit example of a KDF that is compliant with this specification. This will help interoperability and compliance between NIST's standardized KDFs and IETF applications using HKDF as defined in the above RFC.

It is true that an interested party can check this compliance (as I did) but
this is not obvious from reading 56C and hence a basis for confusion in the
industry (I can speak from experience...). Indeed, figuring out compliance
requires a careful "lawyery" reading of 800-56C and 800-108.
One can see that HKDF as defined in RFC 5869 is an implementation of the
Extraction-then-Expansion KDF defined in 56C where
-- HMAC-hash is used for the extraction and expansion
-- The expansion part uses feedback mode from SP-800-108 Sec 5.2
   K(i) := PRF (KI, K(i-1) {||[i]2 }|| Label || 0x00 || Context || [L]2)
   where IV is set to empty, the optional counter is included, the 0X00
   separator is excluded, the length parameter L is excluded, and only one
   of Label or Context are included.

The fact that 108 allows this configuration is not easy to verify. For example
the above expression from Sec 5.2 of 108 shows only the counter to be optional
(by putting it under {}) while the other values would seem to be non-optional.
Then in page 12 of 108 it is explicitly said that

> "In the following key derivation functions, the fixed input data is a
> concatenation of a Label, a separation indicator 0x00, the Context, and [L].
> One or more of these fixed input data fields may be omitted..."

so these are indeed optional thus making RFC 5869 compliant with 56C. An
additional element to note is that IV can be empty per page 8 in 108.
(Note: I recommend that in 56C page 12 you change line 10 as follows
 "When concatenating the above encoded fields, or part of them, the length...")

I believe that the IETF and the industry at large will be served by an explicit statement that HKDF as defined in RFC 5869 is an acceptable implementation of the KDFs allowed by 56C (and 108). If 56C is judged inappropriate for such cross reference with RFC 5869, you may consider putting it as part of 800-135.

Issue 2: Mixed hashes?
=====================

In section 6 you mandate the use of the same hash function for extraction and expansion. I agree that this will be the most common configuration but I do not see why others should be "outlawed". As long as the output from the expansion is of length at least as the hash length for the extraction part, the combination could be allowed. Actually, the analytical results presented in my work show that SHA-512 for extraction and SHA-256 for expansion may have benefits even over the use of SHA-512 for both functionalities (the truncation of the output of SHA-512 to 256 bits eliminates information from the attacker and allows for an analysis using milder assumptions on the hash function).

Note that you are allowing AES-256 for extraction and AES-128 for expansion so it is not different in the hash case. (I know that in the CMAC case the use of AES-128 in expansion is forced by the 128-bit output of AES-256; you could have concluded that one should use AES-128 for both but you rightly allowed the mix.)

Let me thank you for documenting this work. Hopefully, it will encourage the use of the extract-then-expand approach to KDF in more industry applications.
I believe this will contribute to a better practice of cryptography!

Hugo

Adam Petcher

1) I couldn't find any recommendations on the length of the salt for the HMAC-based extraction procedure. The document should contain this information.

2) The salt used in the extraction procedures is allowed to be a string of zeros. What are the practical implications of using this value as the salt? The document should describe these implications so an implementer will know whether it is worthwhile to sacrifice other desirable system features in order to obtain good salt values.

Adam Petcher